

2025-05-27 10:20:40

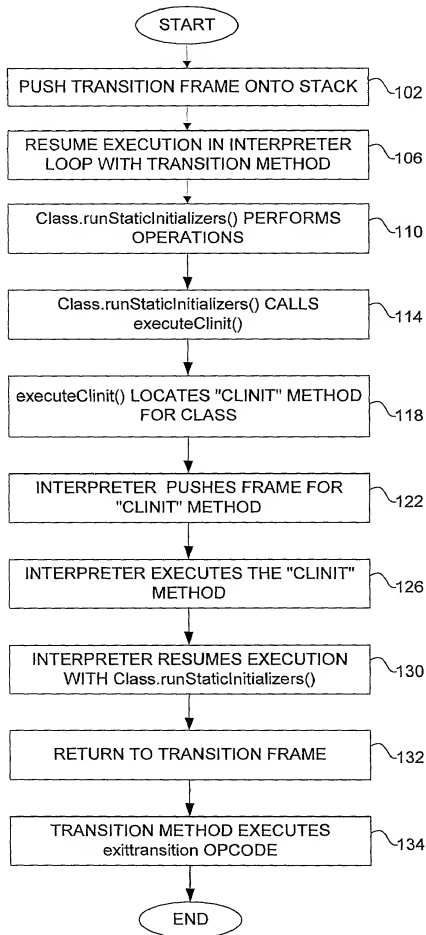


Figure 1

```

private void runStaticInitializers() throws Throwable {
    CVM.disableRemoteExceptions();
    try {

        /* Step 1 - lock the class. */
        synchronized (this) {

            /* Step 2 - if someone is currently initializing this class
             * and it's not our thread, then wait until the class is
             * done initializing. */
            while (checkInitializingFlag(false)) {
                try {
                    wait(0);
                } catch (InterruptedException e) {
                    /* This can't happen since async exceptions are
                     * disabled during static initializers.
                     */
                }
            }

            /* Step 3 and 4 - return success if this thread is
             * initializing the class or if the class is already
             * initialized. */
            if (checkInitializingFlag(true) || checkInitializedFlag()) {
                return;
            }

            /* Step 5 - fail if class is already in an erroneous state. */
            if (checkErrorFlag()) {
                throw new NoClassDefFoundError(getName());
            }

            /* Step 6 - mark the class as "initializing" and unlock it. */
            setInitializingFlag();
        }

        /* Step 7 - run the superclass static initializer. */
        Class superClass = getSuperclass();
        if (superClass != null && !superClass.checkInitializedFlag()) {
            try {
                superClass.runStaticInitializers();
            } catch (Throwable e) {
                synchronized (this) {
                    clearInitializingFlag();
                    setErrorFlag();
                    notifyAll();
                }
                throw e;
            }
        }

        /* Step 8 - run the static initializer. */
        Throwable clinitError = null;
        try {
            CVM.executeClinit(this);
        } catch (Throwable e) {
    
```

Figure 2

1004757-042401

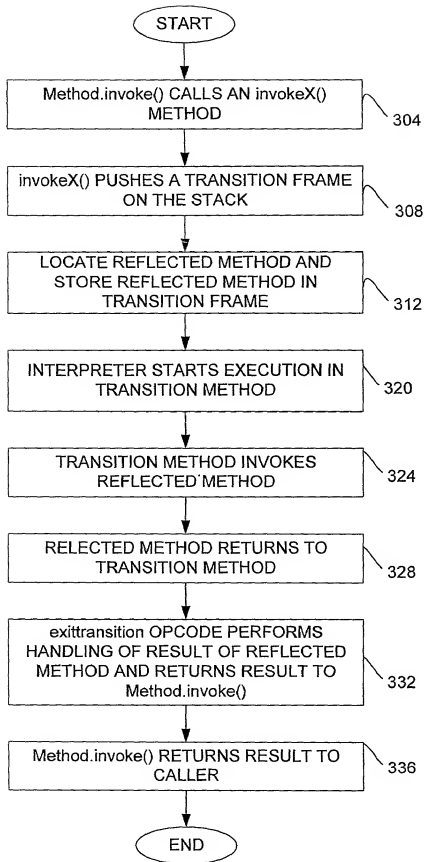


Figure 3

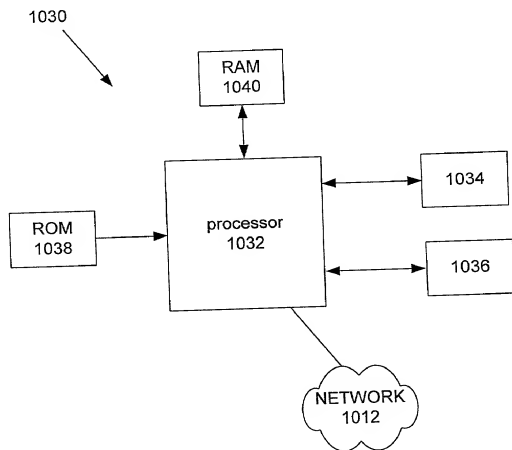


Figure 4

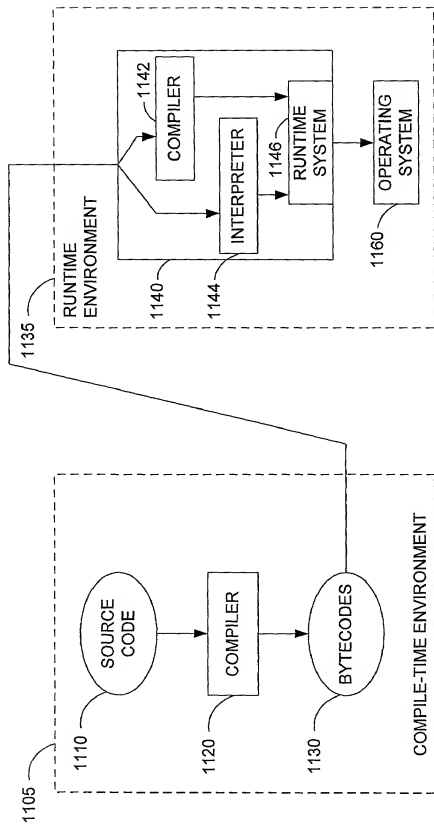


Figure 5